REMARKS

Claims 116-162 are pending in the Application.

Claims 116-162 have been rejected.

Claims 116, 128, 137, 146, and 155 have been amended. No new matter has been added. Support for these amendments can be found, at least, within page 19, lines 7-20 and Figure 7 of the Specification.

Rejection of Claims under 35 U.S.C. § 103(a)

Claims 116-163 stand rejected under 35 U.S.C. § 103(a) as purportedly being anticipated by U.S. Patent No. 5,864,842, issued to Pederson et al. ("Pederson") in view of U.S. Patent No. 6,564,204 issued to Amundsen ("Amundsen"). Applicants respectfully traverse this rejection.

Amended independent Claim 116:

> receiving a first table and a second table, in a computer system;
> generating, using a processor, a set of SQL statements to query the first table and the second table, wherein
>   the first table and the second table are stored in a computer-readable storage medium, and
>   the generating uses a relationship between the first table and the second table to construct the set of SQL statements, and
>   the set of SQL statements comprises SQL statements other than a statement that joins the first and second tables;
> querying the first table using the set of SQL statements to produce a first result set, wherein
>   the querying the first table is performed using the processor;
> querying the second table using the set of SQL statements to produce a second result set, wherein
>   the querying the second table is performed using the processor, and
>   the querying the first table and the querying the second table are performed without joining the first table and the second table; and
> joining, using the processor, the first result set and the second result set to produce a third result set.

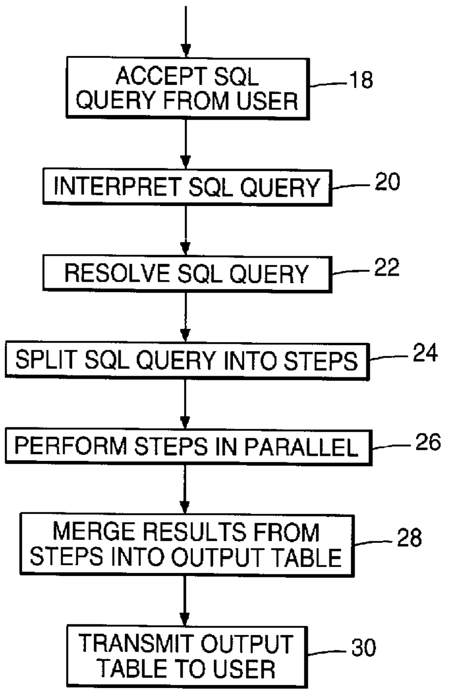
Emphasis added. Independent Claims 128, 137, 146, and 155 recite similar limitations. Applicants respectfully submit that neither Pederson nor Amundsen, alone or in any combination, teach or suggest, at least: (1) receiving a first and second table and generating a set of SQL statements to query the first and second table, where the set of SQL statements comprises SQL statements other than a statement that joins the first and second tables; (2) generating the set of SQL statements using a relationship between the first and second table; (3) querying the first table using the set of SQL statements to produce a first result set, where the querying is performed without joining the first and second tables; and (4) querying the second table using the set of SQL statements to produce a second result set, where both querying operations are performed without joining the first and second tables.

The Office Action cites the following section of Pederson as purportedly disclosing generating a set of SQL statements to query a first and second table, wherein the set of SQL statements comprises SQL statements other than a statement that joins the first and second tables:

> Block 18 represents an SQL query being accepted by the IFP node 12. Block 20 represents the SQL query being transformed by an SQL interpreter executing on the IFP node 12. Block 22 represents the SQL interpreter resolving symbolic names in the query using a data dictionary that contains information about all the databases and tables in the system. <u>Block 24 represents the SQL interpreter splitting the query into one or more "step messages", wherein each step message is assigned to an AMP node 12</u> identified by a hash bucket.
>
> As mentioned above, the rows of the tables are evenly distributed among all AMP nodes 12, so that all AMP nodes 12 can work at the same time on the data of a given table. If a request is for data in a single row, i.e., a prime index request, the IFP node 12 transmits it to the AMP node 12 in which the data resides. If the request is for multiple rows, then the steps are forwarded to all participating AMP nodes 12. Since the database tables are evenly distributed across the DSUs 16 of the AMP nodes 12, the workload of performing the SQL query is balanced between the AMP nodes 12 and DSUs 16.
>
> Block 24 also represents a dispatcher task executed by the IFP node 12 sending the step messages to their assigned AMP nodes 12 via the interconnect network 14.

Pederson 4:25-47 (emphasis added). This passage discusses Pederson's Figure 2, which is a flow chart illustrating the steps necessary for the interpretation and execution of SQL queries. *See* Pederson, 4:14-15. The underlined text in the above-cited passage refers to the splitting of the query detailed in Block 24 of Figure 2, which appears as follows.

FIG. 2


ACCEPT SQL QUERY FROM USER
18
INTERPRET SQL QUERY
20
RESOLVE SQL QUERY
22
SPLIT SQL QUERY INTO STEPS
24
PERFORM STEPS IN PARALLEL
26
MERGE RESULTS FROM STEPS INTO OUTPUT TABLE
28
TRANSMIT OUTPUT TABLE TO USER
30

Pederson, Figure 2. Block 18 represents the acceptance of a SQL query from a user. Block 24 represents Pederson's SQL interpreter splitting the query – the rows of the tables in the query will also be split into sub-portions. *See* Pederson 4:35-36 ("the rows of the tables are evenly distributed among all AMP nodes 12 [of Figure 1], so that all AMP nodes 12 can work at the same time.").

As explained by Pederson, once the query and tables are split and assigned to the AMP node subprocessors, "[e]ach of the sub-queries are then performed independently of one another to create intermediate result tables." Pederson 6:54-56. In other words, the intermediate result tables in Pederson are produced by performing the subquery on the table sub-portions that have been split among the AMP node subprocessors. Thus, Pederson is able to employ parallel processing to perform a star join operation by using a collection of subprocessors to perform hash star join operations on a smaller version of the join operation.

Thus, Pederson is directed toward optimizing SQL operations by using parallel processing. This focus explains why Pederson does not teach or suggest the claimed receiving of a first and second table and generating of a set of SQL statements to query the first and second tables – where the set of SQL statements comprises SQL statements other than a statement that joins the first and second tables. The reason is that Pederson is not concerned with optimizing a SQL operation on a first and second table, by generating a set of SQL statements that do not perform a join on the first and second tables. Pederson is focused on the parallelization of processing join commands, not on the ultimate execution of SQL queries where results of such queries are produced without joins. No cited section of Pederson suggests performing such operations devoid of the use of joins.

The subqueries and table sub-portions generated in Block 24 of Pederson's Figure 2 are executed on the AMP node subprocessors. The AMP node subprocessors, by executing the subqueries on the table sub-portions, produce Pederson's intermediate result tables. *See* Pederson 6:54-56. The subqueries generated in Pederson do not operate on the original data accepted from the user in Block 18, but instead on subsets of that data. By contrast, the generated set of SQL statements in Claim 1 are performed on the first and second tables that are received. The claimed generated set of SQL statements in Claim 1 are not performed on some partitioned sub-portion of the received first and second tables. This is a fundamental distinction in operation, and further explains why Pederson cannot be said to teach or suggest the claimed limitations. This also demonstrates why an ordinary artisan would have no basis for applying Pederson to achieve the method recited in Claim 1. Thus, it cannot be said that Pederson receives a first and second table, nor generates a set of SQL statements to query the first and second table, where the set of SQL statements comprises SQL statements other than a statement that joins the first and second tables.

The Office Action uses the above-cited section of Pederson as purported disclosure of the claimed generation of the set of SQL statements using a relationship between the first and second table. However, it is unclear to Applicants which portion of Pederson is intended to teach this limitation. Thus, Applicants respectfully request guidance regarding which portion of Pederson is intended to teach or suggest that a relationship between first and second tables that are received is used to generate SQL statements to query the first and second table, where the set of SQL statements comprises SQL statements other than a statement that joins the first and second tables. In the absence of such a citation within Pederson, Applicants respectfully submit that Pederson fails to teach or suggest this claimed limitation.

The Office Action purports that the above-cited passage of Pederson discloses "querying the first table using the set of SQL statements to produce a first result set." *See* Office Action, p. 3, citing Pederson 4:35-47. However, as discussed above, Pederson's intermediate result tables are created when the AMP node subprocessors perform the subquery on the table sub-portion. *See* Pederson 6:54-56. By contrast, the claimed first result set is produced by querying the first table using the set of generated SQL statements. Because Pederson's generated subqueries do not operate on the original tables, but rather on a sub-portion of the original tables, Pederson cannot be said to teach or suggest querying the first table using a set of SQL statements to produce a first result set – where the set of SQL statements was generated to query the first table.

Further, Pederson cannot be expected to teach anything like the claimed production of a first result set. Pederson's method is intended to parallelize the execution of a join, and if, instead, the subqueries generated in Pederson were modified to operate on the original input tables, it would undercut the idea behind parallel optimization: that a problem is divided into smaller problems to be solved among the parallel processors. One of ordinary skill in the art would not consider a parallel solution that did not subdivide the original problem. This explains why Pederson cannot be expected to teach or suggest querying the first table using a set of SQL statements to produce a first result set – where the set of SQL statements was generated to query the first table.

The Office Action states, and Applicants agree, that, "Pederson does not disclose querying the second table using the set of SQL statements to produce a second result set wherein the querying the second table is performed using the processor, and the querying the first table and the querying the second table are performed without joining the first table and the second table." Office Action, pp. 3 and 4. As an initial matter, Applicants respectfully submit that the reasons that Pederson does not teach this limitation regarding the production of the second table apply equally to the reasons why Pederson does not teach this limitation regarding the production of the first table.

The Office Action then turns to Amundsen, and posits that Amundsen somehow discloses querying the second table using a set of SQL statements to produce a second result set, where the querying the first and second table are performed without joining the first and second tables. *See* Office Action, p. 4. The Office Action cites the following sections of Amundsen as support:

> Along each order of the tensor are a plurality of discrete coordinates, each coordinate relates to a key value found in the corresponding attribute of one or more tuples. Thus, along the first order of the tensor of FIG. 2B, the coordinates are "Cincinnati, Ohio", "Glendale, Ohio", "Rochester, Minn." and "Washington, D.C.", which are the unique values of the city/state attribute of the represented relation. Along the second order of the tensor of FIG. 2B, the coordinates are "45202", "45246", "55901 ", "20231" and "55906", which are the unique values of the postal code attribute of the represented relation. For future reference, the set of coordinates along an order of a tensor will be know as the "domain" of the tensor along that order.

Amundsen 11:50-60; and:

> Within each plane, the each location corresponds to a value for the "first" name and "last" name attributes. Thus, the coordinates of each three dimensional location corresponds to a unique combination of the "first" name, "last" name and "postal code" attributes. It can be seen that in the plane corresponding to the postal code value "55906", there is a numeric value of "1" at the location corresponding to the tuple ("Lance","Amundson","55906"), indicating that there is a tuple in the relation having this combination of attribute values. All other locations in the in the plane corresponding to the postal code value "55906" having a numeric value of "0", indicating that no other tuples have a postal code value of "55906".
>
> It will be appreciated that a tensor representation can be used for a relation with an arbitrary number of attributes; the tensor representation will have one order for each attribute in the relation.

Amundsen 12:40-55.

As an initial matter, Applicants submit that Amundsen pertains to finding an optimized ordering of performing join operations. The problem that Amundsen is attempting to overcome is that the amount of processing and storage space consumed by a join operation depends on the order in which the tables in the join operation are processed. Given the proper statistics about the RDBMS data being joined, an efficient solution to the join can be found. However, it is important to note that the join operations discussed in Amundsen are simply standard joins – Amundsen is directed to the ordering of the join operations to be performed on the tables in an efficient way, not with presenting any sort of join operation (new or otherwise).

To find an efficient ordering of join operations, Amundsen presents the concept of a "relational tensor." *See* Amundsen 5:35-38. Amundsen's relational tensor concept is an alternative representation of data in a RDBMS, and this new storage representation is used to gather statistics about the RDBMS data. This statistical data can then be used to more efficiently perform various database operations, including joins. *See* Amundsen 5:25-29, 5:45-47, and 15:23-30.

In other words, a relational tensor in Amundsen is derived from the data within an RDBMS – not from a query or from the tables input into the query. The relational tensor information is derived from the RDBMS as a whole. As an example, Amundsen explains how the two dimensional relational tensor in Figure 2B is derived from the RDBMS table in Figure 2A, and the three dimensional relational tensor is derived from the RDBMS tables in Figure 2A and 2C. *See* Amundsen 11:9-13:23. Once the relational tensors in Amundsen are derived, a join operation can be efficiently performed.

Given this understanding of a relational tensor in Amundsen, it becomes clear that Amundsen cannot be expected to teach or suggest querying a second table using set of SQL statements to produce a second result set, where the querying the second table is performed without joining the second table with a first table, as claimed. This is at least because Amundsen's relational tensors are used to gather statistics about how to order the steps in the join operation, and once the information is gathered on how to efficiently perform the join operation, the relational tensors are no longer relevant. In other words, the relational tensors have nothing to do with affecting how a join operation functions – a join operation in Amundsen will be performed in the standard way. The relational tensors in Amundsen do not determine that a result set, produced by querying a table, will be produced without performing any join operations. Further, an ordinary artisan would not expect a method for representing data, used before a join is executed, to alter how (or if) the join is (or might be) executed.

Further, the claimed second result set is produced from querying the second table using the generated set of SQL statements. However, the Office Action does not indicate what section of Amundsen is meant to disclose the generated set of SQL statements, which is used to query the second table. Given that the relational tensors generated in Amundsen are not SQL statements, Applicants respectfully submit that Amundsen fails to teach this element Claim 1 (among other limitations). Thus Amundsen cannot be expected to teach or suggest querying the second table using the set of SQL statements to produce a second result set wherein the querying is performed without joining a first and second table.

For at least these reasons, Applicants submit that neither Pederson nor Amundsen, alone or in combination, provide disclosure of all the limitations of independent Claims 116, 128, 137, 146, and 155, and all claims depending therefrom, and that these claims are in condition for allowance. Applicants therefore respectfully request the Examiner's reconsideration and withdrawal of the rejections to these claims and an indication of the allowability of same.

CONCLUSION

In view of the amendments and remarks set forth herein, the Application and the claims therein are believed to be in condition for allowance without any further examination and a notice to that effect is solicited. Nonetheless, should any issues remain that might be subject to resolution through a telephonic interview, the Examiner is invited to telephone the undersigned.

If any extensions of time under 37 C.F.R. § 1.136(a) are required in order for this submission to be considered timely, Applicants hereby petition for such extensions. Applicants also hereby authorize that any fees due for such extensions or any other fee associated with this submission, as specified in 37 C.F.R. § 1.16 or § 1.17, be charged to Deposit Account 502306.

Respectfully submitted,

/ Samuel G. Campbell III /

Samuel G. Campbell III
Attorney for Applicants
Reg. No. 42,381
Telephone: (512) 439-5084
Facsimile: (512) 439-5099